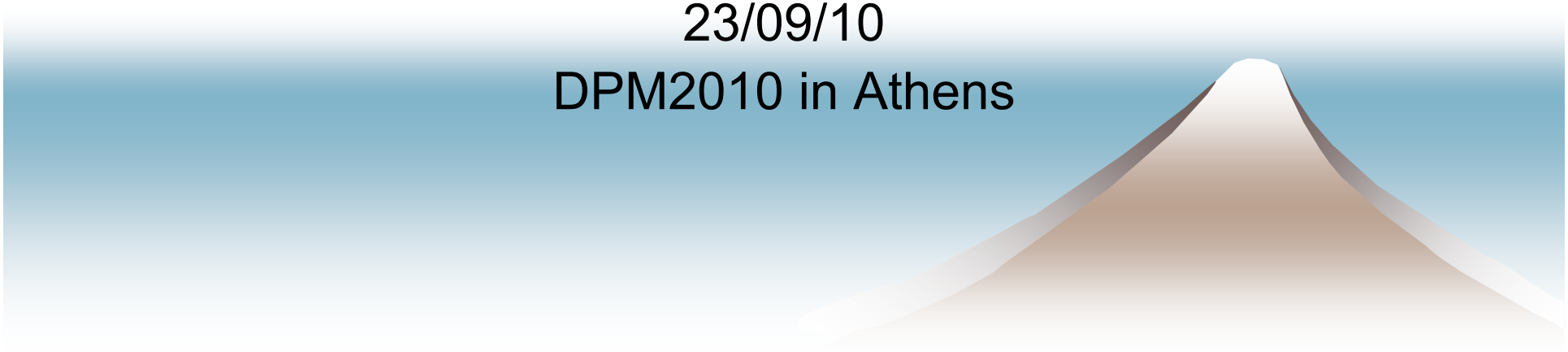
A User-Oriented Anonymization Mechanism for Public Data

Shinsaku Kiyomoto, and Toshiaki Tanaka
KDDI R&D Laboratories Inc.

23/09/10

DPM2010 in Athens



Outline

- Introduction
  - K-anonymity
  - Generalization
- Motivation
- Our algorithm
- Evaluation
- Conclusion

K-Anonymity

- A table T is said to have k-anonymity if and only if each n-tuple of attribute values appears at least k times in T.
- K “same data” exists = an attacker cannot identify a victim from a group of k members.

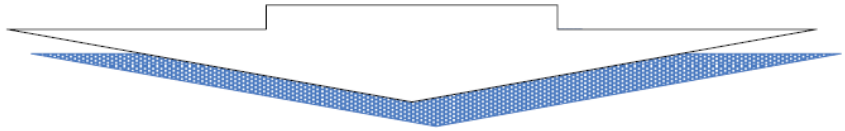
Generalization (1)

- We consider full-domain generalization.
  - Modify (generalize) quasi-identifiers.
  - Modify all values of each attribute as the same level.
  - Tow approach exists;
    - Top-down
    - Bottom-up

Generalization (2)

Original Table

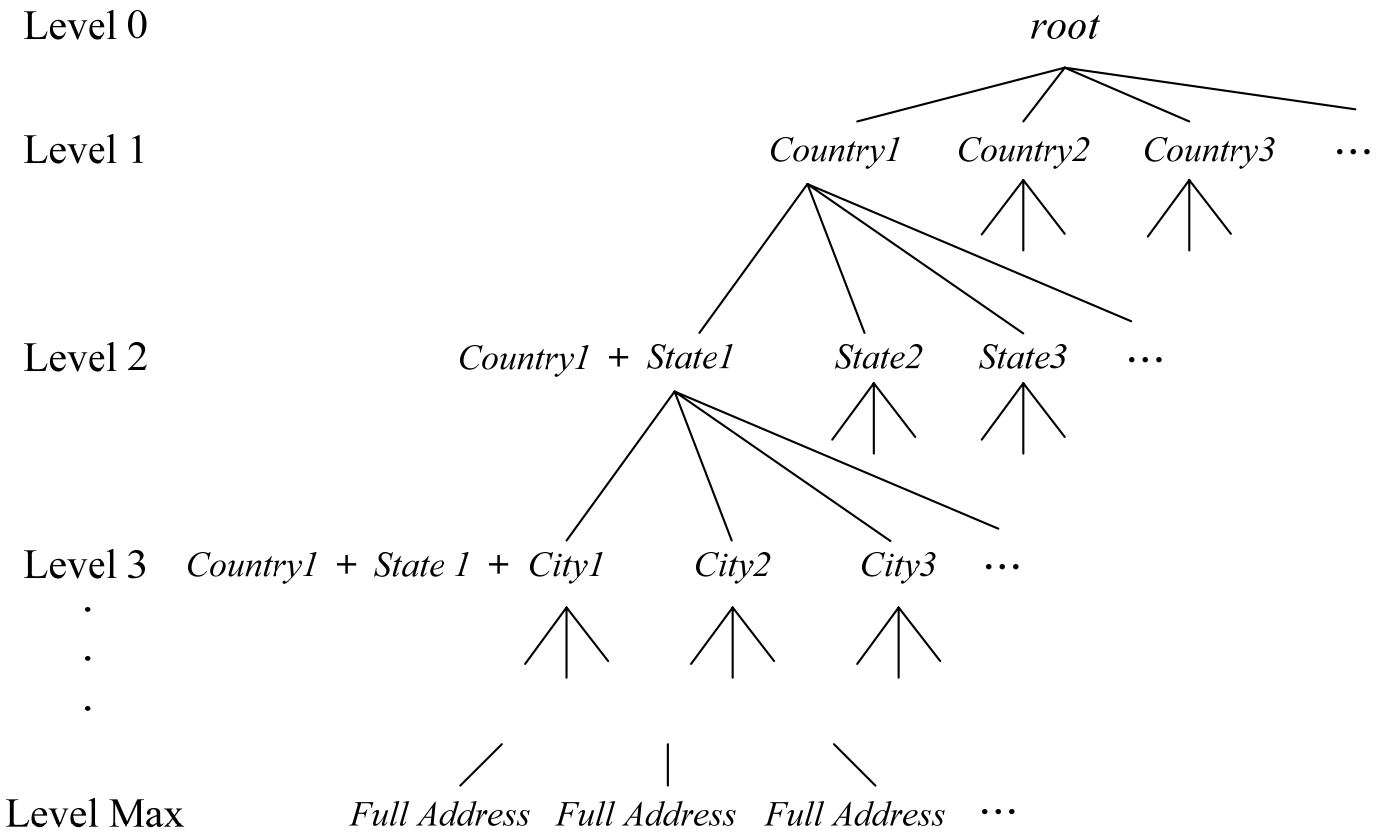
Quasi-Identifiers				Sensitive Information
Birth	Gender	Zip	Nationality	Problem
1985	Male	01243	UK	Shortness of Breath
1985	Male	01242	Italy	Chest Pain
1985	Female	01241	UK	Hypertension
1985	Female	01245	Italy	Hypertension
1984	Male	01234	USA	Chest Pain
1984	Male	01232	USA	Obesity



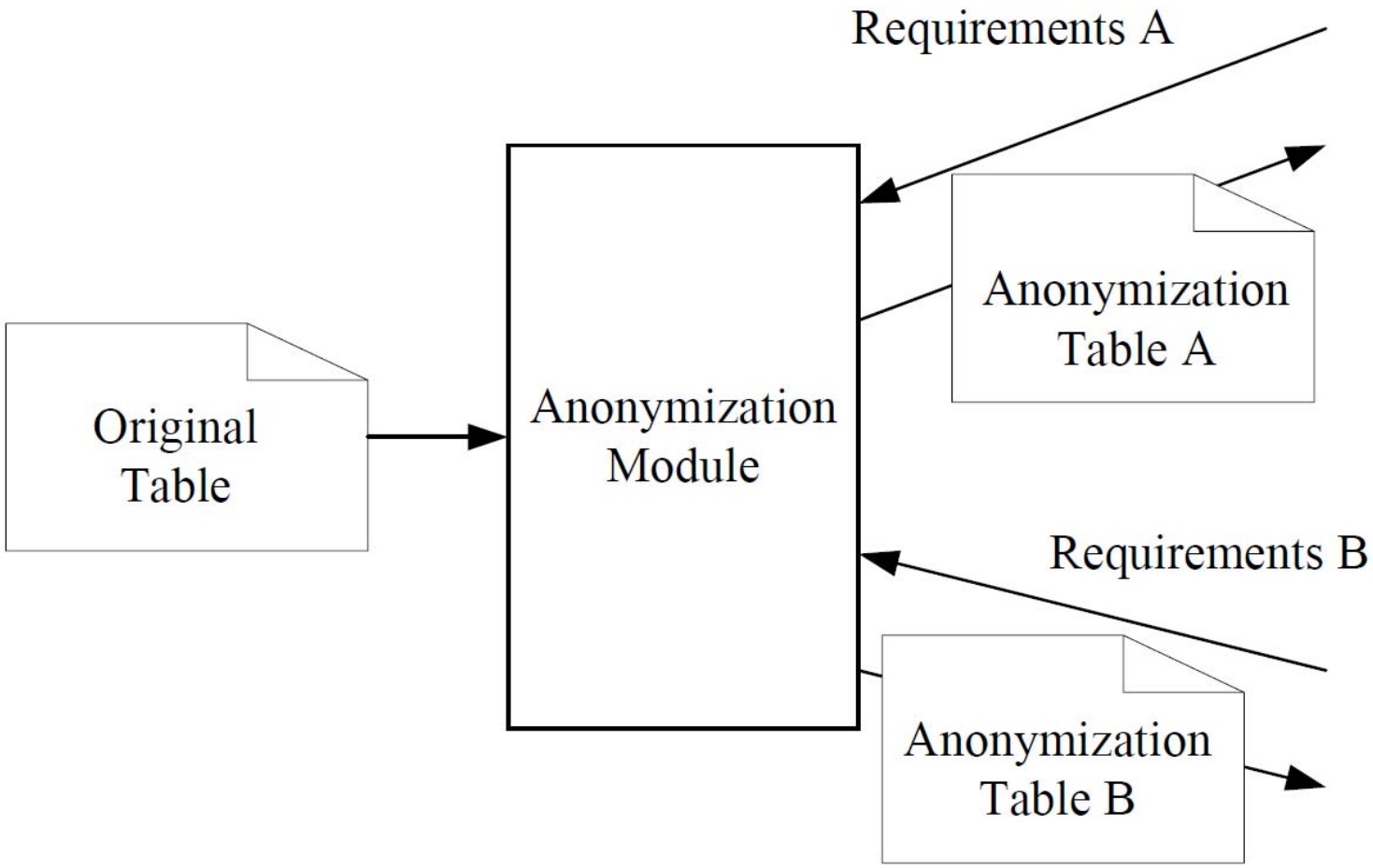
Example of Anonymization Table (k=2)

Quasi-Identifiers				Sensitive Information
Birth	Gender	Zip	Nationality	Problem
1985	Male	0124*	Europe	Shortness of Breath
1985	Male	0124*	Europe	Chest Pain
1985	Female	0124*	Europe	Hypertension
1985	Female	0124*	Europe	Hypertension
1984	Male	0123*	USA	Chest Pain
1984	Male	0123*	USA	Obesity

Hierarchy of attribute values



Service Model (1)



Service Model (2)

- Clients: submit their data to a data holder.
- Data Holder: has an original table, and provides anonymization tables to data users.
- Data Users: are service providers who use anonymization data and provide services to clients or other entities.

Motivation

- The algorithm generalizes the table with consideration of data user's requirements.
  - Generating a table for user's requirements.
  - Fast generation.

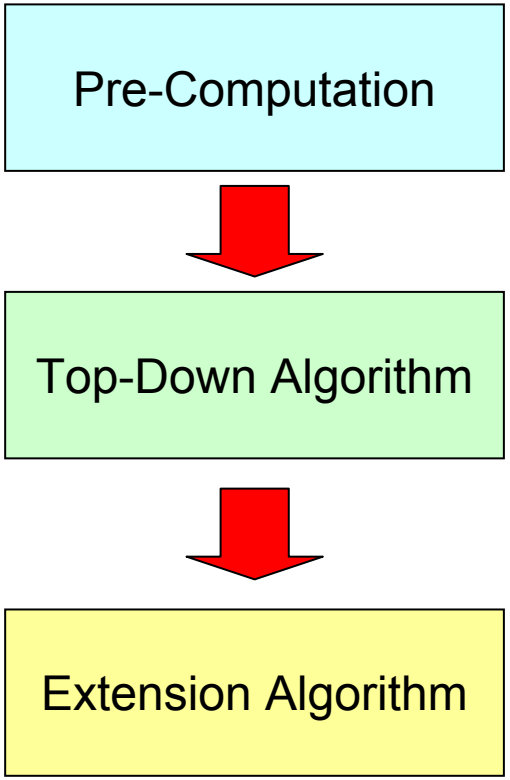
Requirements

- The algorithm considers two requirements
  - Priority. The user can define a positive integer value v_{a^i} for each attribute.

$$(v_{a^1}, v_{a^2}, v_{a^3}, v_{a^4}) = (10, 5, 1, 1)$$

  - Minimum Level. The user can define a minimum level of generalization for each attribute.

Anonymization Process



Find an initial table using single attribute anonymization

Find a k-anonymized table by top-down generalization.

Find an optimal table by top-down and bottom-up generalization.

Pre-Computation

- The algorithm generalizes each attribute to satisfy (k+p)-anonymity.
  - p is a parameter for pre-computation.
- The algorithm checks k-anonymity of the table T. If not, each attribute is generalized once, and the table is checked again.

Top-Down Algorithm

- The algorithm modifies a table using top-down approach and check k-anonymity.
- If k-anonymity is satisfied, the algorithm calculates the score.
- For possible modification of the table, the algorithm calculates scores of modified tables, and chooses the table of the best score.
- The algorithm repeats the modifications and score calculations until no table satisfying k-anonymity exists.

Detail of Basic Algorithm

Pre-Computation

Input: a table T, k, p, H_{a^i}, $w^L_{a^i}$, v_{a^i}, $w^M_{a^i}$ (i=1, ..., n)
Output: T^G, s

// Precomputation:
Sort $(T, v_{a^1}, ..., v_{a^n})$
for $i = 1$ **to** n **do**
  $w_{a^i} \leftarrow w^L_{a^i}$
  while Check$(a^i) < k + p$ **do**
    $a^i \leftarrow$ Generalization(a^i, H_{a^i}, w_{a^i})
    $w_{a^i} \leftarrow w_{a^i} - 1$
  end while
end for
while Check$_k(T) < k$ and all $w_{a^i} > w^M_{a^i}$ **do**
  for $i = 1$ **to** n **do**
    if $w_{a^i} \geq w^M_{a^i}$ **then**
      $a^i \leftarrow$ Generalization(a^i, H_{a^i})
      $w_{a^i} \leftarrow w_{a^i} - 1$
    end if
  end for
end while
$T^I \leftarrow T$
$T^G \leftarrow T^I$
$s, s' \leftarrow$ Score(T)

if Check$_k(T) < k$ **then**
  return *failed*
  end
else
  Top-Down Generation

// Top-Down Generation:
while *state* $\neq$ *stop* **do**
  $T' \leftarrow T^G$
  $s' \leftarrow s$
  state $\leftarrow$ *false*
  for $i = n$ **to** 1 **do**
    $T \leftarrow T'$
    $a^i \leftarrow$ De-Generalization(a^i, H_{a^i}, w_{a^i})
    if Check$_k(T) \geq k$ and Score$(T) > s$ **then**
      temp $\leftarrow a^i, w_{a^i} + 1$
      $T^G \leftarrow T$
      $s \leftarrow$ Score(T)
      state $\leftarrow$ *true*
    end if
  end for
  if *state* $=$ *false* **then**
    state $\leftarrow$ *stop*
  end if
  $a^1, ..., a^n, v_{a^1}, ..., v_{a^n} \leftarrow$ *temp*
end while
return T^G, s

Top-Down Algorithm

Functions

- *Sort* : Sorts attributes by user-defined priority.
- *Check* : Calculates a minimum number of group members with the same attribute values.
- *Generalization* : Modifies a table by bottom-up approach.
- *De-Generalization* : Modifies a table by top-down approach.

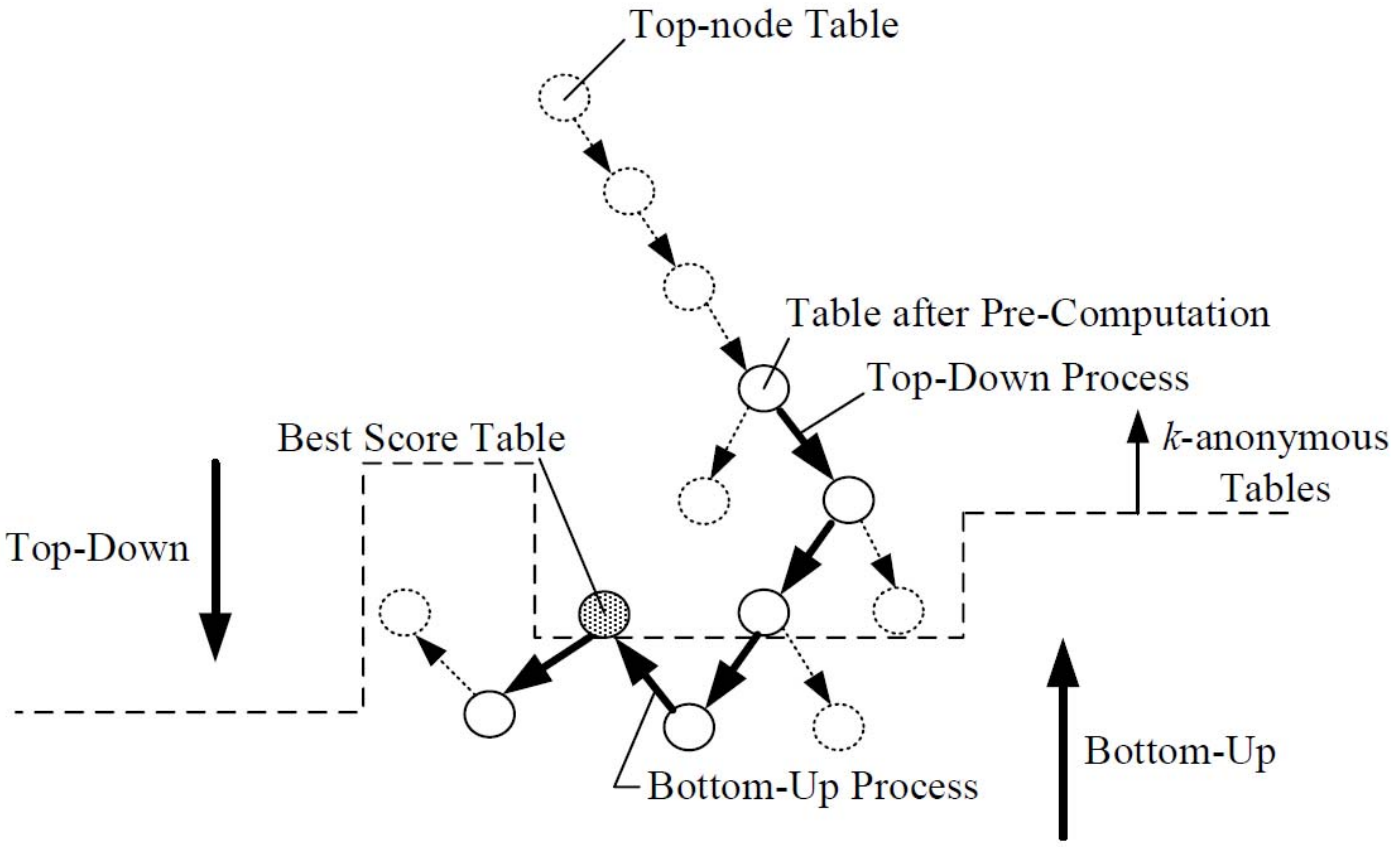
Score Function

- The function calculates the score of the table.

$$S_t = \sum_{\forall a^i} v_{a^i} \cdot e_{a^i}$$

e_{a^i} is a number for the type of values of an attribute a^i

Extension of the Algorithm (1)



Extension of the Algorithm

- Search an optimal table that has higher score than previous.
- Use bottom-up and top-down alternately.
- The algorithm stops where no table has a score higher than the current best score.

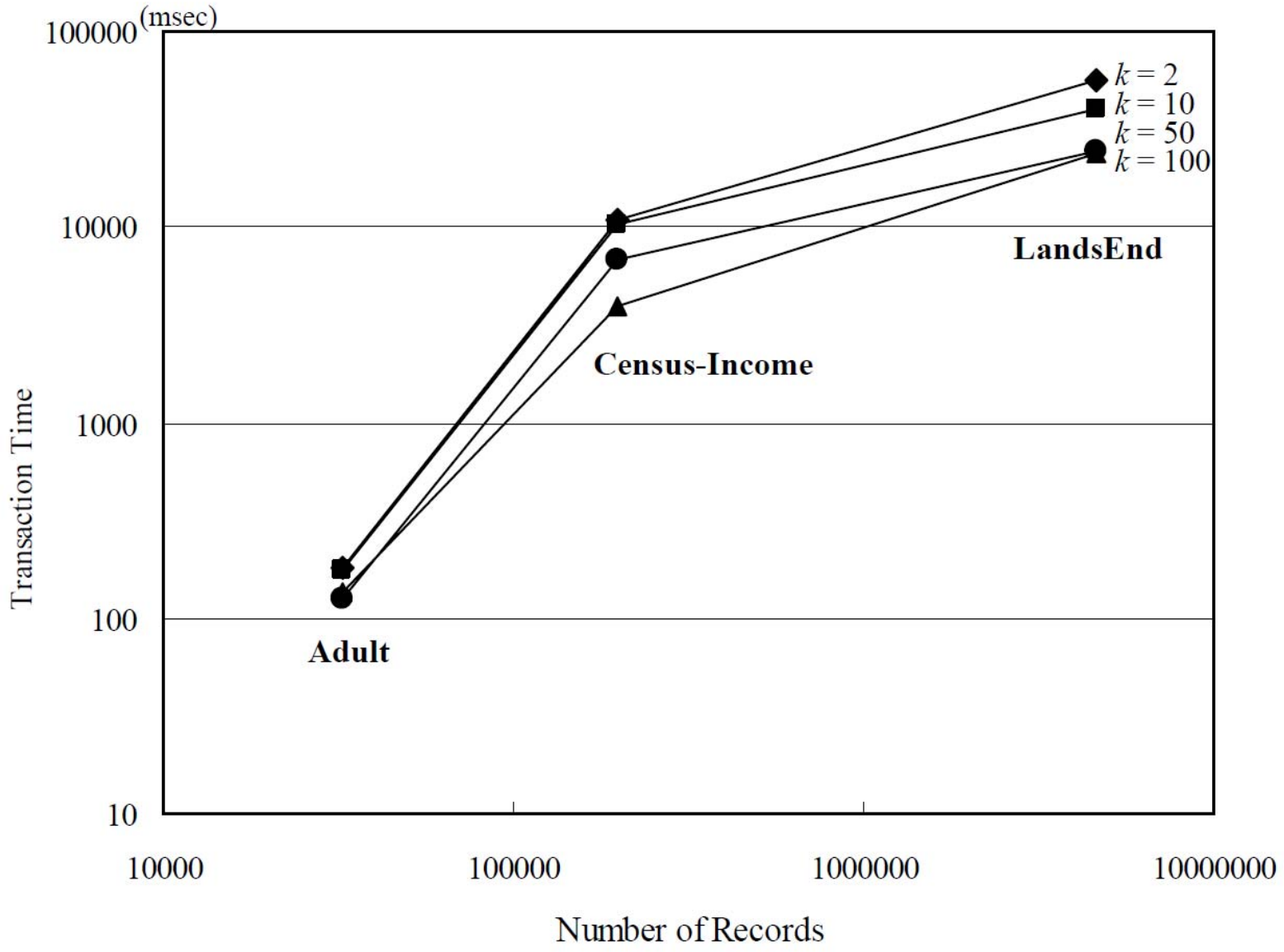
Detail of Extension

Input: a table T^G, k, s, H_{a^i}, $w^L_{a^i}$, v_{a^i}, $w^M_{a^i}$ (i=1, ..., n)
Output: T^G, s

---

repeat
// 2nd Top-Down:
$T' \leftarrow T^G$
$s' \leftarrow s$
$state \leftarrow false$
for $i = n$ **to** 1 **do**
  $T \leftarrow T'$
  $a^i \leftarrow$ De-Generalization(a^i, H_{a^i}, w_{a^i})
  if Score$(T) > s'$ **then**
    $temp \leftarrow a^i, w_{a^i} + 1$
    $T^T \leftarrow T$
    $s^T \leftarrow$ Score(T)
    $state \leftarrow true$
  end if
end for
if $state = false$ **then**
  $state \leftarrow stop$
  return T^G, s
else
// Bottom-Up:
while Check$_k(T) < k$ **do**
$T' \leftarrow T^T$
$state \leftarrow false$
$a^1, ..., a^n, v_{a^1}, ..., v_{a^n} \leftarrow temp$
  for $i = 1$ **to** n **do**
    $T \leftarrow T'$
    if $w_{a^i} > w^M_{a^i}$ **then**
      $a^i \leftarrow$ Generalization(a^i, H_{a^i}, w_{a^i})
    end if
    if Check$_k(T) \geq k$ and Score$(T) > s$ **then**
      $temp \leftarrow a^i, w_{a^i} - 1$
      $T^G \leftarrow T$
      $s \leftarrow$ Score(T)
      $state \leftarrow true$
    end if
  end for
  if $state = true$ **then**
    $a^1, ..., a^n, v_{a^1}, ..., v_{a^n} \leftarrow temp$
  else
  for $i = 1$ **to** n **do**
    $T \leftarrow T'$
    if $w_{a^i} > w^M_{a^i}$ **then**
      $a^i \leftarrow$ Generalization(a^i, H_{a^i}, w_{a^i})
    end if
    if Score$(T) > s'$ **then**
      $T^T \leftarrow T$
      $temp \leftarrow a^i, w_{a^i} - 1$
      $s' \leftarrow$ Score(T)
      $state \leftarrow true$
    end if
  end for
  if $state = false$ **then**
    $state \leftarrow stop$
    return T^G, s
  end if
end while
until $state = stop$

Evaluation Results (1)



Evaluation Results (2)

	No. of Records	k	No. of Check Nodes	Transaction Time	Best Score
Adult (Proposed)	32,561	2	52	226 ms	49
Adult (Top-Down)	32,561	2	598	2207 ms	49
LandsEnd (Proposed)	500,000	50	13	2083 ms	1516
LandsEnd (Top-Down)	500,000	50	540	91186 ms	1516

Summary of Evaluation

- Performance
  - Less than 200ms for 32561 records, 14 attributes.
  - Feasible transaction time.
- Comparison with a conventional top-down method.
  - The best score is the same as that of the conventional.
  - The number of check nodes is 1/10 – 1/40.

Conclusion

- Our Anonymization algorithm
  - Generates a suitable table with feasible transaction time.
- Future work
  - Evaluation using real data.
  - Optimization method for system parameters.
  - Improve the score function.
  - Apply to other generalization schemes.

- Thank you very much for your attention.